

HPC on OpenStack

Review of the our Cloud Platform Project

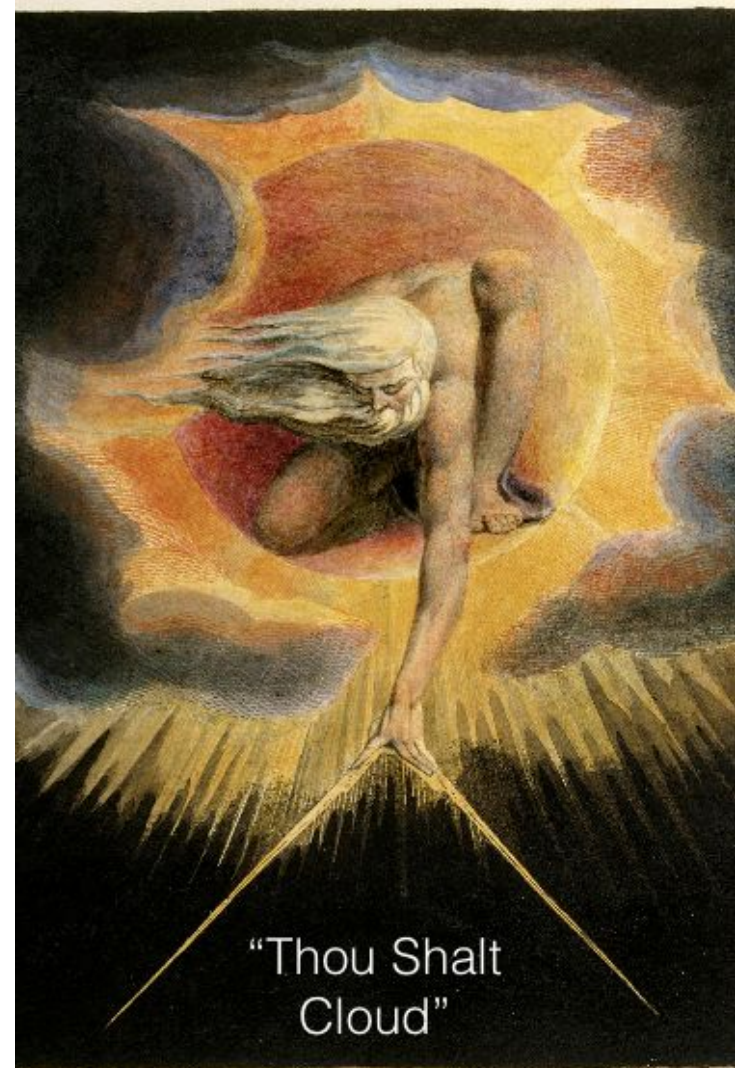
*Petar Foraj, Erich Bingruber, Uemit Seren
Post FOSDEM tech talk event @UGent 04.02.2019*

Agenda

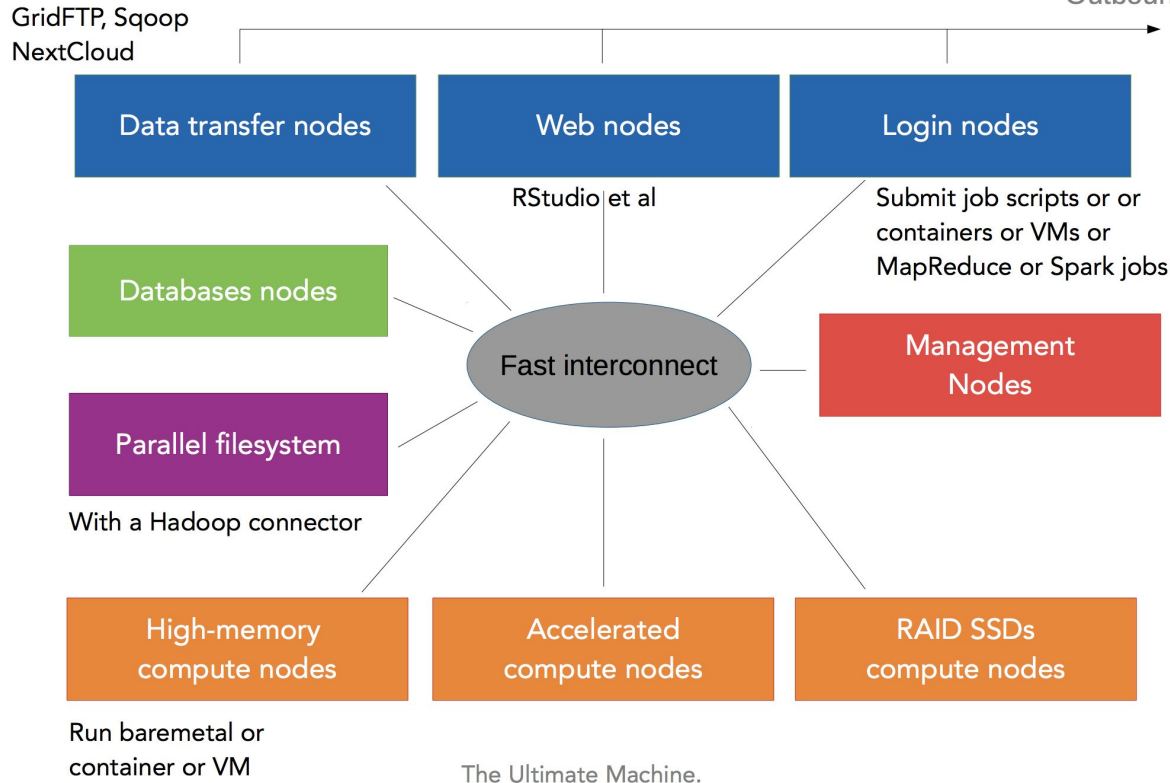
Who We Are & General Remarks (Petar Forai)

Cloud Deployment and Continuous Verification
(Uemit Seren)

Cloud Monitoring System Architecture (Erich
Birngruber)



The “Cloudster” and How we’re Building it!



Shamelessly stolen from Damien François Talk --
“*The convergence of HPC and BigData*
What does it mean for HPC sysadmins?”

Who Are We

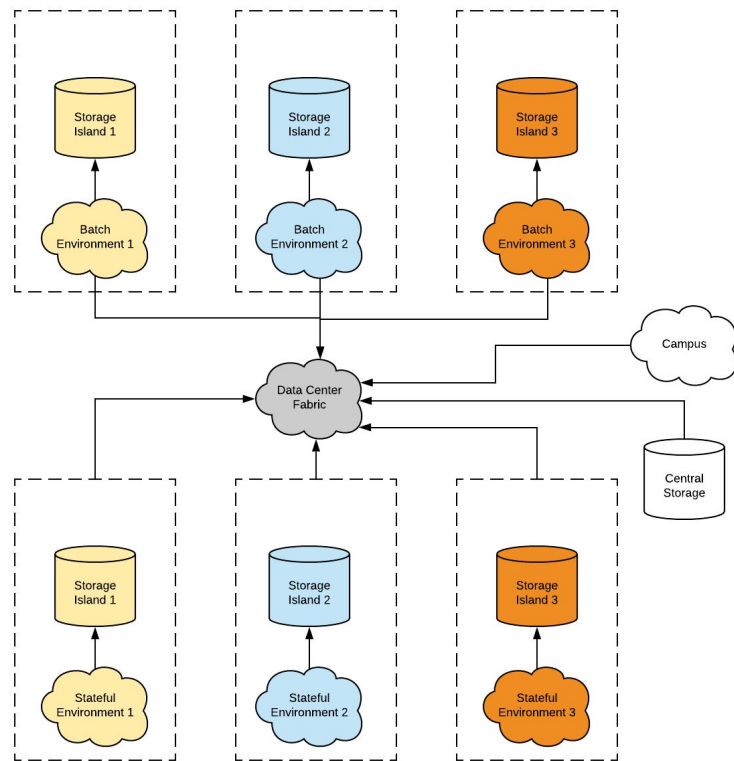
- Part of Cloud Platform Engineering Team at molecular biology research institutes (IMP, IMBA, GMI) located in Vienna, Austria at the Vienna Bio Center.
- Tasked with delivery and operations of IT infrastructure for ~ 40 research groups (~ 500 scientists).
- IT department delivers full stack of services from workstations, networking, application hosting and development (among many others).
- Part of IT infrastructure is delivery of HPC services for our campus
- 14 People in total for everything.

Vienna Bio Center Computing Profile

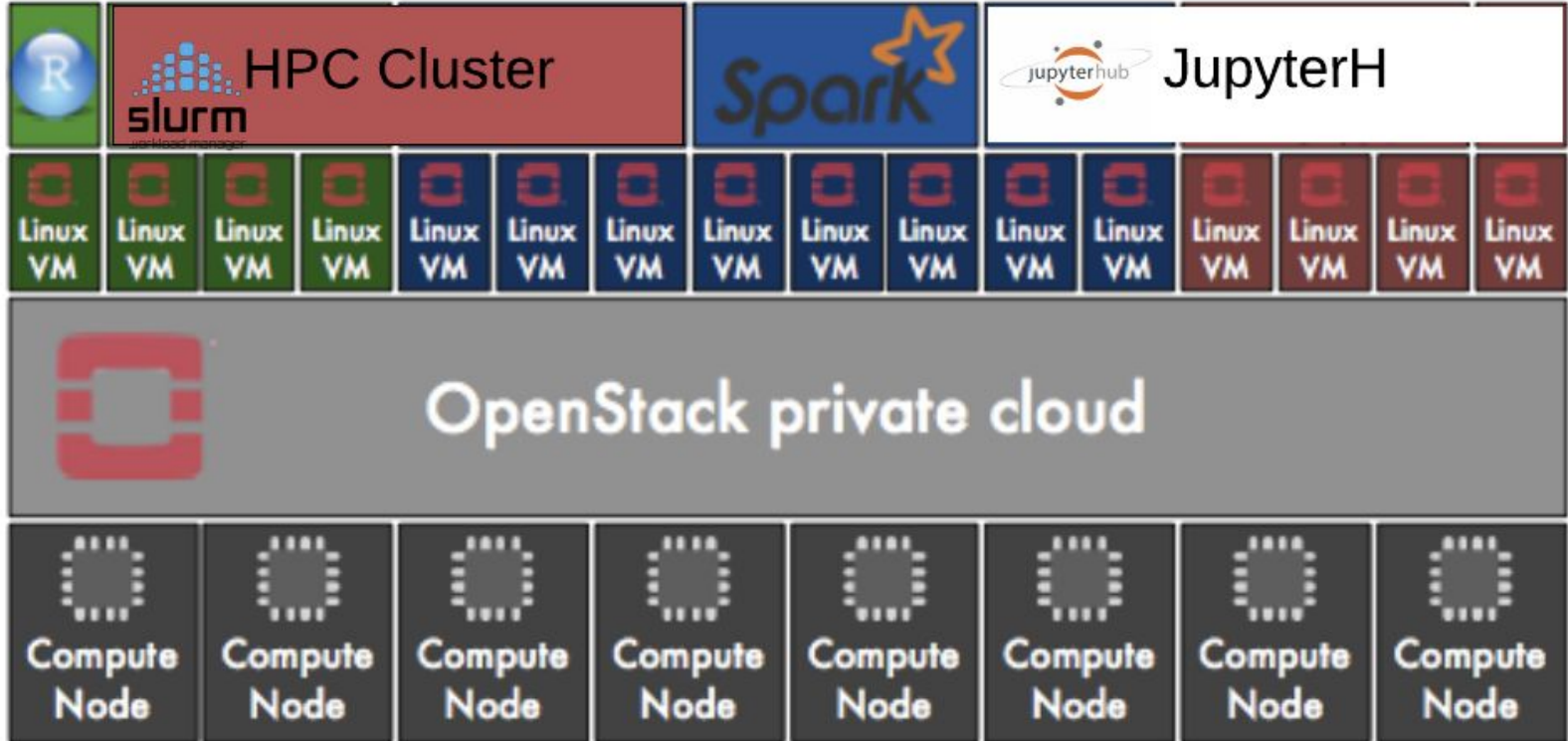
- Computing infrastructure almost exclusively dedicated to bioinformatics (genomics, image processing, cryo electron microscopy, etc.)
- Almost all applications are data exploration, analysis and data processing, no simulation workloads
- Have all machinery for data acquisition on site (sequencers, microscopes, etc.)
- Operating and running several compute clusters for batch computing and several compute clusters for stateful applications (web apps, data bases, etc.)

What We Currently Have

- Siloed islands of infrastructure
- Cant talk to other islands, can't access data from other island (or difficult logistics for users)
- Nightmare to manage
- No central automation across all resources easily possible



What We're Aiming At



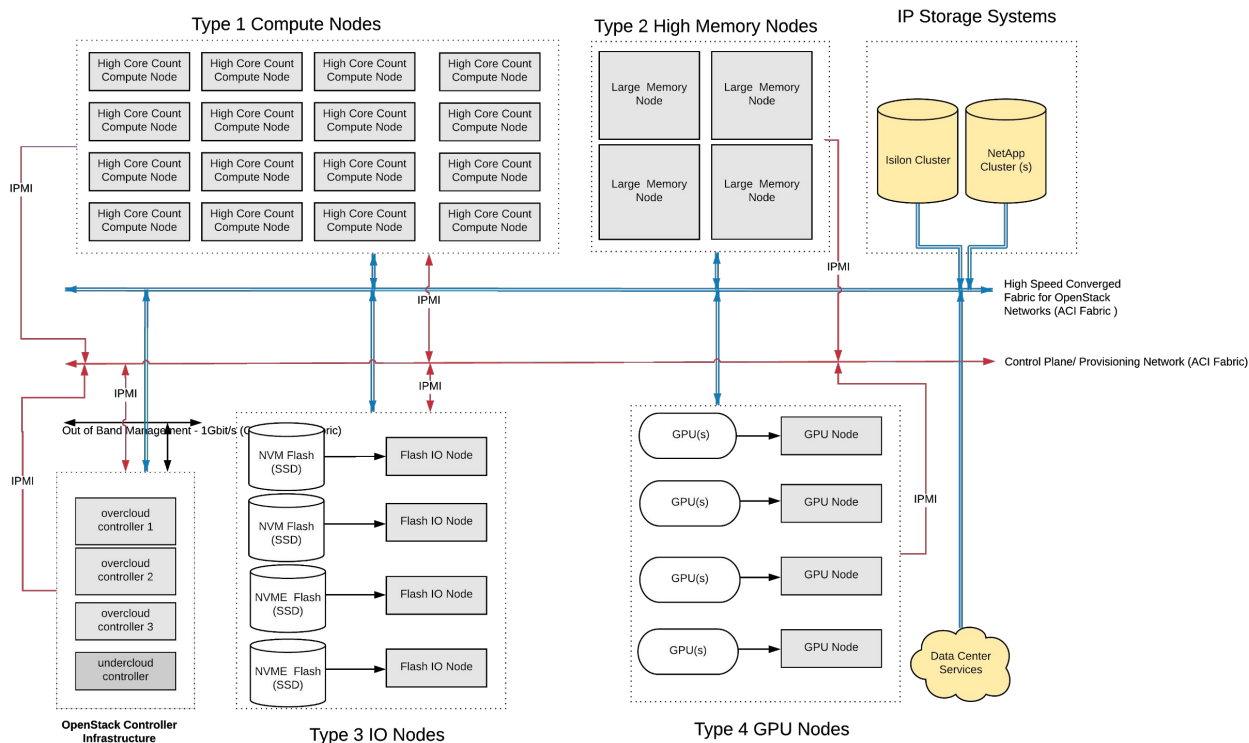
Meet the CLIP Project

- OpenStack was chosen to be evaluated further as platform for this
- Setup a project “CLIP” (Cloud Infrastructure Project) and formed project team (4.0 FTE) with a multi phase approach to delivery of the project.
- Goal is to implement not only a new HPC platform but a software defined datacenter strategy based on OpenStack and deliver HPC services on top of this platform
- Delivered in multiple phases

Tasks Performed within “CLIP”

- Build PoC environment to explore and develop understanding of OpenStack (~ 2 months)
- Start deeper analysis of how OpenStack (~ 8 months)
 - Define and develop architecture of the cloud (understand HPC specific impact)
 - Develop deployment strategy and pick tooling for installation, configuration management, monitoring, testing
 - Develop integration into existing data center resources and services
 - Develop understanding for operational topics like development procedures, upgrades, etc.
 - Benchmark
- Deploy production Cloud (~ 2 months and ongoing)
 - Purchase and install hardware
 - Develop architecture and pick tooling for for payload (HPC environments and applications)
 - Payload deployment

CLIP Cloud Architecture Hardware



- Heterogeneous nodes (high core count, high clock, high memory, GPU accelerated, NVME)
- First phase ~ 100 compute nodes and ~ 3500 Intel SkyLake cores
- 100GbE SDN RDMA capable Ethernet and some nodes with 2x or 4x ports
- ~ 250TB NVMe IO Nodes ~ 200Gbyte/s

HPC Specific Adaptations

- Tuning, Tuning, Tuning required for excellent performance
 - NUMA clean instances (KVM process layout)
 - Huge pages (KSM etc.) setup
 - Core isolation
 - PCI-E passthrough (GPUs, NVME, ...) and SR-IOV (esp. for networking) crucial for good performance

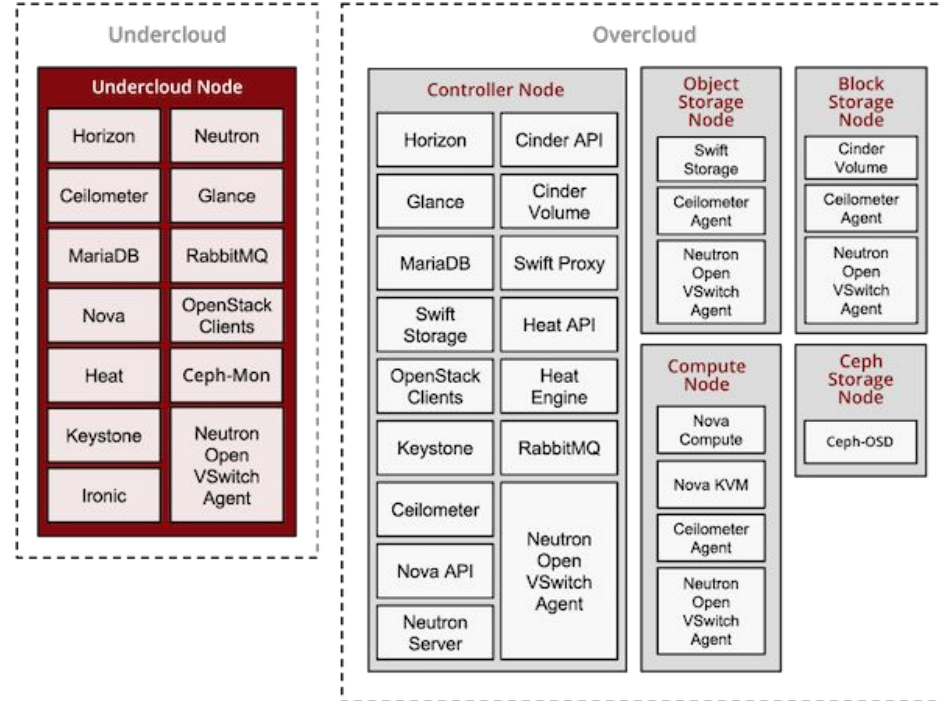
Lessons Learned

- OpenStack is *incredibly* complex
- OpenStack is not a product. It is a framework.
- You need 2-3 OpenStack environments (development, staging, prod in our case) to practice and understand upgrades and updates.
- Out of box experience and scalability for certain OpenStack subcomponents is not optimal and should be considered more of a reference implementation
 - Consider plugging in real hardware here
- Cloud networking is really hard (especially in our case)

Deployment and Cloud Verification

Deployment and Cloud Verification

- Red Hat OpenStack (OSP) uses the upstream “TripleO” (OpenStack on OpenStack) project for the OpenStack deployment.
- Undercloud (Red Hat terminology: Director) is a single node deployment of OS using puppet.
- The Undercloud uses various OpenStack projects to deploy the Overcloud which is our actual cloud where payload will run
- TripleO supports deploying a HA overcloud
- Overcloud can be installed either using a Web-GUI or entirely from the CLI by customizing yaml files.



Deployment and Cloud Verification

The screenshot displays the Red Hat OpenStack Platform Director interface. At the top, the navigation bar includes "RED HAT OPENSTACK PLATFORM DIRECTOR", "Deployment Plan", and "Nodes". A "Register Nodes" dialog is open, showing options to "+ Add New" or "Upload From File", and a "Node Detail" section with a "General" tab.

The main content area shows the "overcloud" deployment progress. The steps are:

- 1 Prepare Hardware**: Includes a "+ Register Nodes" button.
- 2 Specify Deployment Configuration**: Includes a link to "Edit Configuration".
- 3 Configure Roles and Assign Nodes**: Shows "1 Nodes available to assign". A grid of role cards is visible: Block Storage (0 nodes assigned), Controller (0 nodes assigned), Compute (0 nodes assigned), and Object Storage (0 nodes assigned). Each card has an "Assign Nodes" button. The "Controller" card is highlighted with a blue arrow.
- 4 Deploy**: Shows a green success message: "Deployment succeeded Stack CREATE completed successfully".

Below the progress steps, "Overcloud information:" is displayed, including "Overcloud IP address: 10.12.148.155".

The "Controller Role" configuration panel is open, showing tabs for "Parameters", "Services", and "Network Configuration". The "Services" tab is active, listing various services with their OS::TripleO::Services:: prefixes. The "OS::TripleO::Services::Timezone" service is highlighted with a blue arrow. To the right, the configuration details for "Timezone" are shown, with a dropdown menu set to "UTC". Below this, "DefaultPasswords" and "EndpointMap" are shown as empty JSON objects. The "ServiceNetMap" is also shown as an empty JSON object. A "Time" table is visible on the right side of the configuration panel, listing a series of timestamps: -24T07:00:08Z.

Deployment and Cloud Verification

- Web GUI is handy to play around but not so great for fast iterations and Infra as code.
→ **Disable the Web UI and deploy from the CLI.**
- TripleO internally uses *heat* to drive *puppet* that drives *ansible* `¯_(ツ)_/¯`
- We decided to use *ansible* to drive the TripleO OpenStack deployment.
- Deployment split in 4 phases corresponding to 4 git repos:
 - a. *clip-undercloud-prepare*: Ansible playbooks that run on a bastion VM to prepare and install the undercloud using PXE and kickstart.
 - b. *clip-tripleo* contains the customized yaml files for the TripleO configuration (storage, network settings, etc)
 - c. *clip-bootstrap* contains ansible playbooks to initially deploy or update the overcloud using the configuration in the clip-tripleo repo
 - d. *clip-os-infra* contains post deployment customizations that are not exposed through TripleO or very cumbersome to customize

Deployment and Cloud Verification

- TripleO is slow because Heat → Puppet → Ansible !!
 - Small changes require “stack update” → 20 minutes (even for simple config stanza change and service restart).
- Why not move all customizations to ansible (clip-os-infra) ? Unfortunately not robust :-(
 - Stack update (scale down/up) will overwrite our changes
 - → services can be down
- Let's compromise:
 - Iterate on different customizations using ansible
 - Move finalized changes back to *TripleO*
- Ansible everywhere else !
 - *clip-aci-infra*: prepare the networking primitives for the 3 different OpenStack environments
 - Move nodes between environments in the network fabric

Deployment and Cloud Verification

- We have 3 different environments (dev, staging and production) to try out updates and configuration changes. We can guarantee reproducibility of deployment because we have everything as code/yaml, but what about software packages ?
- To make sure that we can predictably upgrade and downgrade we decided to use Red Hat Satellite (Foreman) and create Content Views and Life Cycle Environments for our 3 environments

The screenshot displays the Red Hat Satellite web interface. At the top, the navigation bar includes 'RED HAT SATELLITE' and various menu items like 'IMPIMBA', 'Monitor', 'Content', 'Containers', 'Hosts', 'Configure', 'Infrastructure', and 'Red Hat Insights'. The main content area shows the 'ccv-clip' Content View details, including a 'Versions' tab. A table lists the versions of the content view, with columns for Version, Status, Environments, Content, Description, and Actions.

Version	Status	Environments	Content	Description	Actions
Version 17.0	Promoted to Production (2018-12-10 10:04:06 UTC)	Library Development Staging Production	26852 Packages 4070 Errata (644 ▲ 2754 ● 672 ☐)		Promote
Version 16.0	Promoted to Production (2018-11-29 04:10 UTC)		26665 Packages 4038 Errata (639 ▲ 2727 ● 672 ☐)	Now for real with RHEL 7.6	Promote
Version 14.0	Promoted to Production (2018-11-27 16:00:55 UTC)		24264 Packages 3656 Errata (594 ▲ 2467 ● 595 ☐)	Bumped ovosp12 to 3.0	Promote
Version 13.0	Published (2018-08-30 13:35:19 UTC)		24825 Packages 3640 Errata (590 ▲ 2455 ● 595 ☐)	bumped ovos-rhel-7server version to 2.0	Promote
Version 8.0	Promoted to Staging (2018-08-31 09:57:00 UTC)		23766 Packages 3593 Errata (581 ▲ 2424 ● 588 ☐)	Added python2-networking-sfc	Promote
Version 7.0	Promoted to Development (2018-08-20 11:39:38 UTC)		23765 Packages 3593 Errata (581 ▲ 2424 ● 588 ☐)	Updated ov-clip to include supervisor, f5ltd and python-mel3	Promote
Version 4.1	Incremental Update (2018-08-28 14:08:32 UTC)		23780 Packages 3594 Errata (581 ▲ 2425 ● 588 ☐)		Promote
Version 4.0	Promoted to Development (2018-08-20 10:50:36 UTC)		23762 Packages 3593 Errata (581 ▲ 2424 ● 588 ☐)	Upgraded ov-clip content view to include Cisco ACI RBMs	Promote
Version 3.0	Promoted to Development (2018-07-31 12:11:49 UTC)		23745 Packages 3593 Errata (581 ▲ 2424 ● 588 ☐)	Added ov-clip content view	Promote
Version 1.0	Promoted to Development (2018-07-31 11:48:30 UTC)		23744 Packages 3593 Errata (581 ▲ 2424 ● 588 ☐)	Initial version	Promote

Deployment and Cloud Verification

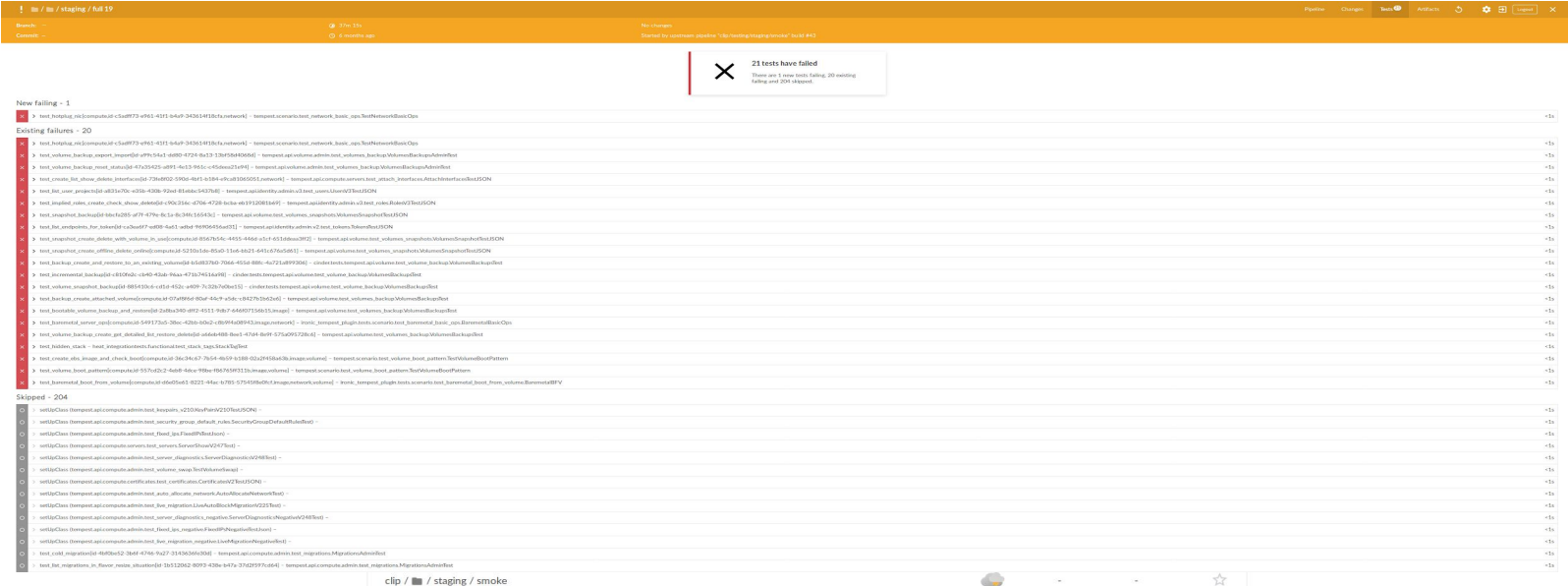
- While working on the deployment we ran into various known bugs that are fixed in newer versions of OSP. To keep track of the workaround and the status of those bugs we use a dedicated JIRA project (CRE)

The screenshot shows a JIRA Kanban board for the 'CLIP CRE Kanban' project. The board is organized into columns: To Do, In Progress, Review, Waiting, Workaround, and Done. Each column contains several issues (CRE-25 to CRE-23) with their titles, descriptions, and status indicators.

Issue ID	Title	Description	Status
CRE-25	Controller runs out of disk space	Backlog	To Do
CRE-28	Cinder generic NFS driver (sillon) can't create a backup from a volume with a snapshot	Backlog	To Do
CRE-33	Nova-Cinder - restart of nova compute containers leads to recursive ownership takeover of /var/lib/nova	Backlog	To Do
CRE-34	Neutron-sriov-agent NeutronSriovNumVFs on mib4	Backlog	To Do
CRE-20	ACI Integration TripleO Templates shipped by Cisco are not Pike compatible	In Progress	In Progress
CRE-29	TripleO should create cinder default volume type	In Progress	In Progress
CRE-30	ACI TripleO templates support only one global interface name for uplink interface	Open	Open
CRE-35	enable cert checking for infoblox api calls	Open	Open
CRE-37	Selinux is preventing OpenStack from launching multiqueue-enabled instances	Open	Open
CRE-27	Cinder's generic NFS driver (sillon) can't use Nova's os-assisted-volume-snapshots	In Review	In Review
CRE-3	nova - multiqueue tap device - increase no. of queues	Waiting	Waiting
CRE-10	swift_rsync container is in restarting loop	Waiting	Waiting
CRE-12	sensu client container inconsistent IP reporting	Waiting	Waiting
CRE-13	collected turbostat plugin parsing error	Waiting	Waiting
CRE-14	rsyslog Fluentd integration	Waiting	Waiting
CRE-19	ACI Integration Module CBP OpFlex Interface MTU is hard coded to 1600 bytes	Waiting	Waiting
CRE-31	CBP Mapping Driver does not support shared external groups/policies	Waiting	Waiting
CRE-5	installing of CiscoACI Puppet RPM via TripleO	Workaround	Workaround
CRE-6	openstack network agent delete not working	Workaround	Workaround
CRE-7	Overcloud nodes are registered in Satellite with shortname instead of fqdn	Workaround	Workaround
CRE-8	Enabling fencing for control plane requires two deployments	Workaround	Workaround
CRE-9	extra binds to collect container	Workaround	Workaround
CRE-15	Overcloud deployment fails with "No valid host was found. There are not enough hosts available."	Workaround	Workaround
CRE-17	Heptapod logs and overcloud validation shows reds down	Workaround	Workaround
CRE-21	cinder - sillon nfs locking issue	Workaround	Workaround
CRE-23			
CRE-1	all the things	Resolved	Resolved
CRE-2	make hostnames great again	Resolved	Resolved
CRE-4	Typo in heat template for PublicVirtualFixedIPs	Resolved	Resolved
CRE-11	Designate multi controller coordination backend	Resolved	Resolved
CRE-16	Overcloud deployment fails because of RHEL registration error	Resolved	Resolved
CRE-18	List of container images in overcloud_images.yaml are missing some containers	Resolved	Resolved

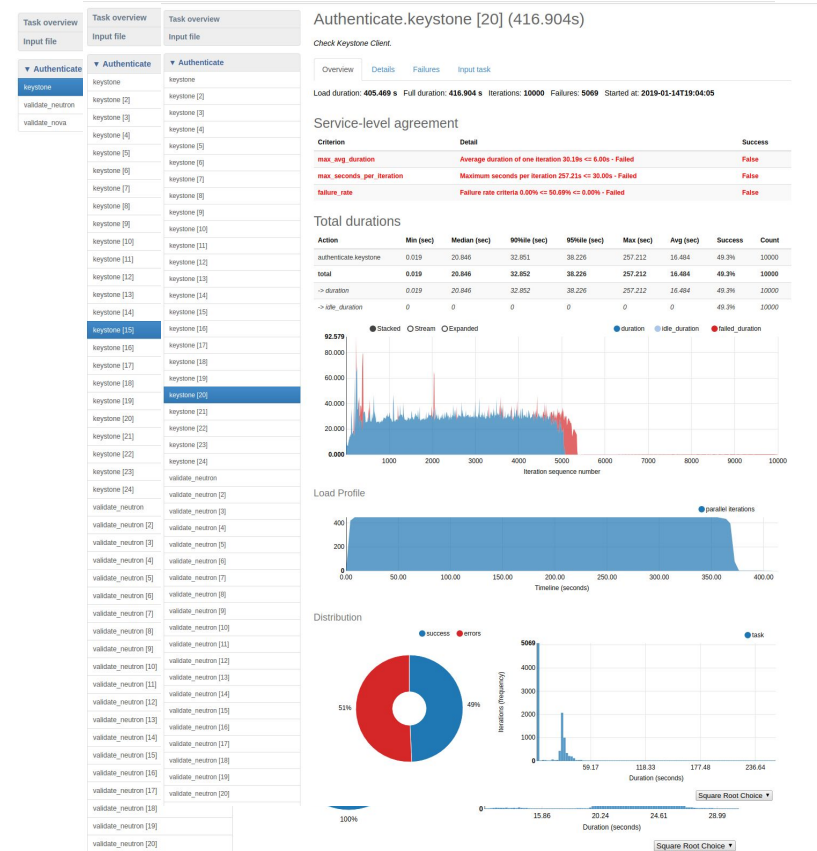
Deployment and Cloud Verification

- How can we make sure and monitor that the cloud works during operations ?
- We leverage OpenStack's own tempest testing suite to run verification against our deployed cloud.
- First smoke test (~ 128 tests) and if this is successful run full test (~ 3000 tests) against the cloud.



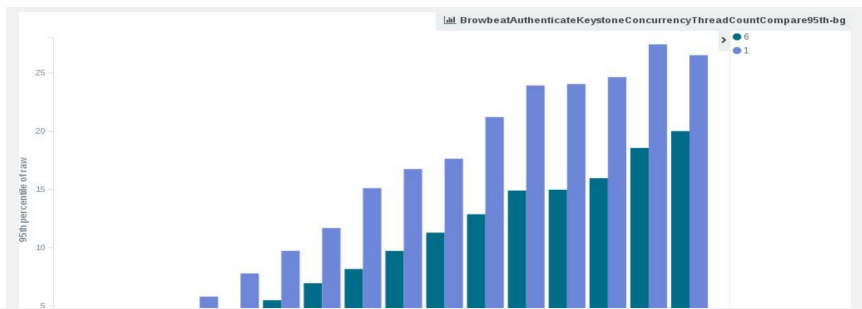
Deployment and Cloud Verification

- Ok, the Cloud works but what about performance ? How can we make sure that OS performs when upgrading software packages etc ?
- We plan to use *Browbeat* to run *Rally* (control plane performance/stress testing), *Shaker* (network stress test) and *PerfkitBenchmarker* (payload performance) tests on a regular basis or before and after software upgrades or configuration changes



Deployment and cloud verification

- Grafana and Kibana dashboard can show more than individual rally graphs:

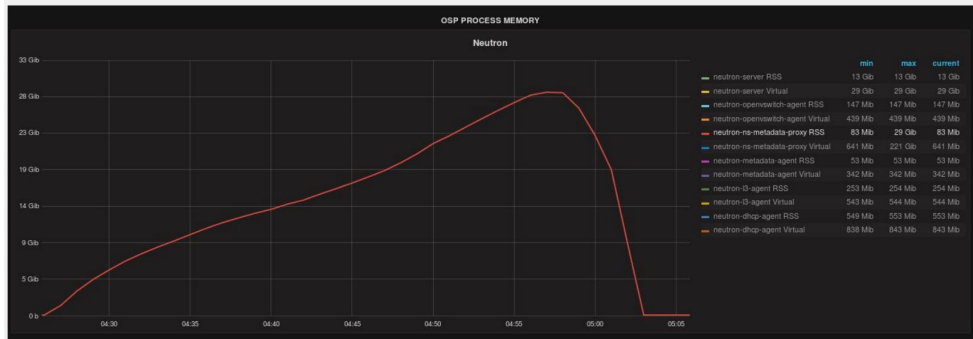


- Browbeat can show differences between settings or software versions:

Scrolling through Browbeat 22 documents...

Scenario	Action	conc.	times	0b5ba58c	2b177f3b	% Diff
create-list-router	neutron.create_router	500	32	19.940	15.656	-21.483
create-list-router	neutron.list_routers	500	32	2.588	2.086	-19.410
create-list-router	neutron.create_network	500	32	3.294	2.366	-28.177
create-list-router	neutron.create_subnet	500	32	4.282	2.866	-33.075
create-list-port	neutron.list_ports	500	32	52.627	43.448	-17.442
create-list-port	neutron.create_network	500	32	4.025	2.771	-31.165
create-list-port	neutron.create_port	500	32	19.458	5.412	-72.189
create-list-subnet	neutron.create_subnet	500	32	11.366	4.809	-57.689
create-list-subnet	neutron.create_network	500	32	6.432	4.286	-33.368
create-list-subnet	neutron.list_subnets	500	32	10.627	7.522	-29.221
create-list-network	neutron.list_networks	500	32	15.154	13.073	-13.736
create-list-network	neutron.create_network	500	32	10.200	6.595	-35.347

UUID	Version	Build	Number of runs
938dc451-d881-4f28-a6cb-ad502b177f3b	queens	2018-03-20.2	1
6b50b6f7-acae-445a-ac53-78200b5ba58c	ocata	2017-XX-XX.X	3



Deployment and cloud verification

Lessons learned and pitfalls of OpenStack/TripleO:

- OpenStack and TripleO are complex with many moving parts. → **Have a dev/staging environment to test the upgrade and pin the software versions with Satellite or Foreman.**
- Upgrades (even minor ones) can break the cloud in unexpected ways. Biggest pain point was to upgrade from OSP11 (non-containerized) -> OSP12 (containerized).
- Containers are no free lunch. You need a container build pipeline to customize upstream containers to add fixes and workarounds.
- TripleO gives you a supported out of the box installer for HA OpenStack with common customizations. Non-common customizations are hard because of rigid architecture (heat, puppet, ansible mixed together). TripleO is moving more towards ansible (config download)
- *“Flying blind through clouds is dangerous”*: Make sure you have a pipeline for verification and performance regression testing.
- Infra as code (end to end) is great but requires discipline (proper PR reviews) and release management for tracking workarounds and fixes.

Cloud Monitoring System Architecture

Monitoring is Difficult

Because it's hard to get these right

- The information
 - At the right time
 - For the right people
- The numbers
 - Too few alarms
 - Too many alarms
 - Too many monitoring systems
- The time
 - doing it too late

Monitoring: What We Want to Know

- Logs: as structured as possible → Fluentd
 - syslog (unstructured)
 - OpenStack logs (structured)
- Events → RabbitMQ
 - OpenStack RPCs
 - high-level OpenStack interactions
 - CRUD of resources
- Status → Sensu
 - polling: is the service UP?
 - Publish / subscribe
 - modelling service dependencies
- Metrics → Collectd
 - time series, multi dimensional
 - performance metrics

Monitoring: How We do it

- Architecture

- Ingest endpoints for all protocols
- Buffer for peak loads
- Persistent store
 - Structured data
 - timeseries

- Dashboards

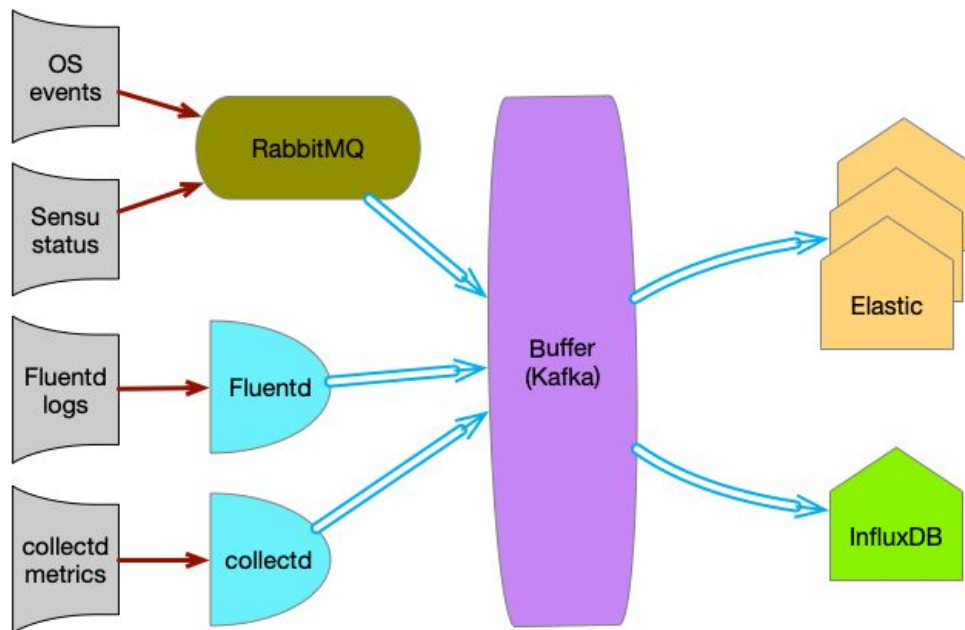
- Kibana, Grafana
- Alerta to unify alarms

- Integration with deployment

- Automatic configuration

- Service catalog integration

- Service owners
- Pointers to documentation



Monitoring: Outlook

- What changes for cloud deployments
 - Lifecycle, services come and go
 - Services scale up and down
 - No more hosts
- Further improvements
 - infrastructure debugger (tracing)
 - Stream processing (improved log parsing)
 - Dynamically integrate call-duty / notifications / handover
 - Robustness (last resort deployment)

Thanks!